# Implications of the Java™ Language on Computer-Based Patient Records

Daniel Pollard, Ezra Kucharz MS, W. Edward Hammond Ph.D.
Division of Medical Informatics
Duke University Medical Center
Durham, NC

*The growth of the utilization of the World Wide Web (WWW) as a medium for the delivery of computer-based patient records (CBPR) has created a new paradigm in which clinical information may be delivered. Until recently the authoring tools and environment for application development on the WWW have been limited to Hyper Text Markup Language (HTML) utilizing common gateway interface scripts. While, at times, this provides an effective medium for the delivery of CBPR, it is a less than optimal solution. The server-centric dynamics and low levels of interactivity do not provide for a robust application which is required in a clinical environment. The emergence of Sun Microsystems' Java language is a solution to the problem. In this paper we examine the Java language and its implications to the CBPR. A quantitative and qualitative assessment was performed. The Java environment is compared to HTML and Telnet CBPR environments. Qualitative comparisons include level of interactivity, server load, client load, ease of use, and application capabilities. Quantitative comparisons include data transfer time delays. The Java language has demonstrated promise for delivering CBPRs.*

## INTRODUCTION

The growth of the World Wide Web (WWW) has facilitated an emergence of prototypical computer-based patient records (CBPR) on the Internet. The WWW is a client-server mechanism which uses Hyper Text Transfer Protocol (HTTP) to transfer information. The interfaces, database engines, and architectures of each WWW based CBPR are unique[1]. Most are programmed in Hyper Text Markup Language (HTML) and utilize common gateway interface (CGI) scripts to access medical data stored in database servers. Problems with this paradigm include: server-centric dynamics, low levels of data and interface interactivity, minimal interface representations and limited types of active client processes.

Server-centric dynamics are present when there is a heavy reliance on the server to perform most computing functions. Such dynamics are a significant problem for two reasons. First, large institutions with many clients tend to generate heavy loads on the central processing units (CPU) of their servers. Secondly, most computations of dynamic data representations must also be performed on the server, further increasing the load.

Interface limitations of HTML constrain the interactivity of a HTML based CBPR to low levels. HTML was designed to present static information and images with hyperlinks as the form of interactivity. The advanced interface tools of image maps and forms are essentially fancy forms of hyperlinks. Any other manipulation or modification to the data representation must be sent to and processed by the server. For these reasons, an interactive HTML interface is very server intensive.

In terms of client CPU utilization, HTML fails as well. The only active client processes which occur that utilize HTML are those unmodifiable processes running in the browser (e.g. Netscape). These include helper applications and recently some client-side image maps.

The limitations of HTML create an interesting paradox. To this point HTML has been the only method of creating interfaces and representing information on the WWW. While these interfaces and representations provide, at times, an effective paradigm for CBPR, they do not provide the true level of interactivity necessary for the utilization of the WWW as a clinical tool.

After reviewing some of the WWW CBPR projects in the literature[1,2,3,4], it became apparent that a new medium of programming and representation was necessary to accomplish the task of implementing a WWW CBPR that would be used in a clinical setting. A WWW based CBPR system must have a real-time interactive interface, be platform independent, utilize an internet-based programming language, and utilize client-based processors. This new paradigm of Internet communication is

0195—4210/96/$5.00 

characteristic of Sun Microsystems' Java programming language and environment. The purpose of this paper is to examine the Java language for utilization in a WWW CBPR.

## BACKGROUND

Java has its origins in an effort to create software for consumer electronics. Initially, development was done in the C++ programming language. Along the way, many problems were encountered trying to use C++ for these small, reliable, portable, distributed, real-time systems. C++ had problems in the area of processor independence and was unable to provide reliable and safe execution. In fact, designing a new language became a better alternative; thus Java was incarnated. Several object oriented programming languages (SmallTalk, Objective C, Eiffel) and a thread management system (Xerox Park's Cedar/Mesa) influenced the design and architecture of Java[5]. As the Java language matured, the consumer electronics market became less attractive for Sun. Eventually, Sun exited the consumer electronics market, and Java became a technology without a home.

The ability to develop secure, distributed, network-based end-user applications in Java has made it a perfect tool for the Internet. Java applications can be securely delivered to and run on any hardware and software platform[6]. Furthermore, Java applications consume minimal system resources and can be dynamically extended.

### Java is Similar to C

Java can be considered a leaner and meaner cousin of the ANSI C programming language. Some have termed Java the C programming language of the Internet. To better understand Java, it is important to see how it differs from C[7].

First of all, Java does not support several of the major data types found in C. These are pointers, structures and unions. Java passes all arrays and objects by reference without an explicit pointer type. This feature circumvents many programming and security problems. First, pointers can not be created which point to undefined locations in memory. Secondly, many programming errors result from the misuse of pointers. Finally, without pointers, security is improved because access to locations in memory can not be forged. Structures or unions are also not supported. Java instead replaces their functionality with the object oriented constructs of classes and interfaces. Another significant difference between Java and C is the removal of the preprocessor, header files (.h files), define statements (#define) and type definitions (typedef). These constructs make it quite difficult to read and understand C code. Without these, Java code is easier to understand and therefore easier to modify and re-use. It should be noted that there are other minor differences, but for the sake of space they are not covered in this document[5].

### Java is able to Run Existing C Code

Java has the capability to call existing programs. This is accomplished by creating a Java method whose implementation is written in an external programming language. Currently, the only external programming language which can be linked to Java methods is C.

### Applications vs. Applets

Two types of programs can be written in Java: applications and applets. Java applications are stand alone programs which can be executed by a Java interpreter. HotJava, an Internet browser, is a prime example of a stand alone Java application. Java applets, as the name implies, are small Java applications. They are designed to be executed in the environment of a web browser. Specifically, they are designed to be called by an <APPLET> tag in a HTML document, downloaded from a Web server across the Internet and run locally in conjunction with a web page. The most popular Java enabled web browser is the Netscape Navigator 2.0. The navigator contains a Java interpreter and therefore can run Java applets. Presently, the Java enabled Netscape Navigator is available on many platforms including Windows 95, Windows NT, Macintosh PowerPC, HP-UX (Hewlett Packard UNIX), SGI IRIX (Silicon Graphics UNIX), OSF/1, SunOS 4.1, and Sun Solaris 2.3 and 2.4. Netscape anticipates shipping Java support for Windows 3.1, Macintosh, and AIX in the second quarter of 1996[8].

### Current CBPRs Using Java

An extensive literature and WWW search for the utilization of the Java language in on-line medical applications revealed that little application development has occurred to date. The Gamelan website proved to have the most extensive list of medical Java applets and applications[9]. At the time of authoring this paper, seven applets were available. Most were educational in nature and included topics such as genetic mapping and The Visible Human

Project. However, one applet from Los Alamos National Laboratory was a radiology interface for the Telemed Virtual Patient Record System. The Telemed Virtual Patient Record System is a project with the purpose of creating a national imaging and medical records system for radiologists[10]. This prototype applet utilizes a frontal plane x-ray view of the chest region to navigate through cross sectional x-rays of the same region. No other Java CBPR has been located in the literature or WWW. This scarcity can be attributed to the only recent release of the Java Developers Kit and Java extensions to Netscape's WWW browser. The pending release of Java programming tools and environments such as Symantec's Cafe, Metroworks' CodeWarrior Java Extension, Borland's C++ (5.0) Java Extension, and Rogue Wave Software's JFactory will increase the growth of Java applet development. These tools will simplify and streamline Java programming.

## METHODS

To evaluate a Java applet as a means of medical information delivery, a set of medical information was defined and presented using three different delivery methods: a live telnet connection, HTML pages, and a Java applet (see Figures 1,2, and 3).

First, a set of medical information was defined. Three screens of data were identified (demographics, encounters and problems) in an existing electronic medical record system (TMR)[11]. Next, HTML web pages and a Java applet were programmed to display the exact same information.

The Java applet and the HTML pages were served from a 60MHz Pentium running Windows NT 3.51 directly connected to the Duke University Medical Center network. The TMR telnet server was a VAX/VMS machine also connected directly to the Duke network. The client which accessed all three resources was a 75 MHz Pentium running Windows95 connected to the Internet via a dedicated 56Kbit connection. Netscape Navigator 2.0 (for HTML and Java) and WinQVT/Net 3.9 (for Telnet) were the software packages used.

First, the time of the initial download was measured. For the telnet connection, this was the time taken to connect to the host. For the HTML implementation, it was the time to load up the home page. For the applet, it was the time to load and start the applet. The reload feature of Netscape was used to load the HTML and Java pages without client caching effects.

*Figure 1. Telnet session with Encounter List.*

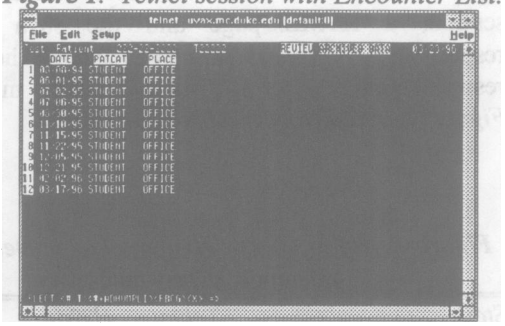



*Figure 2. HTML page with Encounter List.*

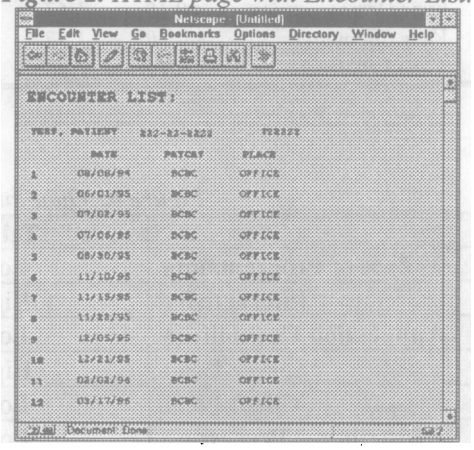
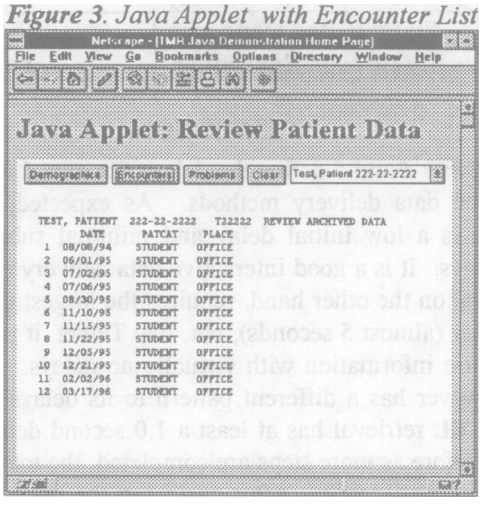



*Figure 3. Java Applet with Encounter List*




Next, the time to view each piece of information in the set was measured. For Telnet, this was the time between choosing a menu choice and the display of the information. For HTML, this was the time between clicking on the hyperlink and the display of the secondary page. For the applet, this was the time between pressing an applet button and the display of the information.

*Figures 1, 2* and *3* show the screens of a Telnet session, a HTML page and the Java applet respectively. *Figures 4 and 5* show the time response for each of the three delivery methods. *Figure 6* qualitatively compares the three.

***Figure 4.*** *Time Delays of Different Data Delivery Methods( in seconds)*

| *Step* | *Telnet* | *HTML* | *Java* |
|---|---|---|---|
| *A. Initial Download* | *1.17* | *1.32* | *4.74* |
| *B. View Demographics.* | *<0.10* | *1.67* | *<0.10* |
| *C. View Encounters* | *<0.10* | *1.58* | *<0.10* |
| *D. View Problems* | *<0.10* | *1.26* | *<0.10* |

***Figure 5.*** *Graph of Total Time Delay of the Different Data Delivery Methods*

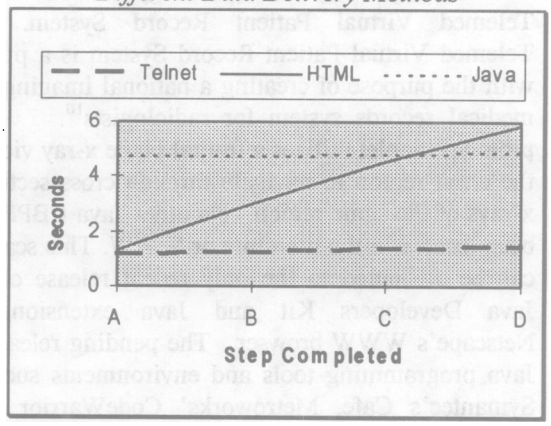



***Figure 6.*** *Qualitative Comparison of Data Delivery Methods*

| | Telnet | HTML | Java |
|---|---|---|---|
| Ease of Programming | Low | High | Low |
| Interactivity | High | Low | High |
| Application Capabilities | Low | Low | High |
| Server Load | High | High | Low |
| Client Load | Low | Low | High |
| Distributed Processing | Low | Low | High |
| Ease of Use (Interface) | Low | High | High |
| Platform Independence | High | High | High |
| Security through Encryption | Low | High | High |

## DISCUSSION

*Figures 4* and *5* define the delays associated with the three data delivery methods. As expected, Telnet shows a low initial delay and minimal subsequent delays. It is a good interactive data delivery method. Java, on the other hand, requires the largest up-front delay (almost 5 seconds), but, like Telnet, it provides future information with virtually no delays. HTML however has a different pattern to its delays. Each HTML retrieval has at least a 1.0 second delay, and therefore as more steps are completed, the total delay increases linearly. After retrieving the third HTML page (after step C), the HTML user has experienced a greater cumulative delay than a Java or Telnet user.

These results suggest that dynamic information with graphical content is best displayed using Java. If graphical content is not required, Telnet is adequate. If the data is graphical yet static, HTML works best.

The qualitative description in *Figure 6* addresses some additional issues. First, ease of programming evaluated the skill needed by a programmer to create the CBPR on the WWW. HTML proved to be the easiest to program of the three. It should be noted that most meaningful HTML solutions incorporate CGI scripts to interface with databases and incorporate user input. The programming of such CGI scripts is a non-trivial task. Both the Java and the Telnet solutions are more complex programming tasks. The difficulties in programming Java pay a worthwhile dividend in the level of interactivity and application capabilities. The user interface and capabilities of Java parallel the functionality of most high level programming languages (including C/C++). The static graphic representations in HTML and the text only displays in Telnet are eclipsed by the robustness of Java.

In terms of distributed processing, Java does well. It utilizes both the client's and the server's CPUs. Telnet and HTML both heavily rely on the server's processor. Telnet is a continuous connection between the client and server in which most of the processing occurs on the server. The client acts

strictly as a terminal. Further problems with Telnet include the requirement for constant availability of the server and the delays in service due to network traffic. HTML uses an asynchronous communication scheme in which the client contacts the server only when additional information is needed. While this reduces the load on the server compared with the Telnet solution, the server remains a processing bottleneck. Every time the client attempts to get information from the server it must wait for a new connection to be opened. Java utilizes a different scheme; it downloads the entire applet at once and then the applet becomes self contained on the client. It should be noted that with the present version of HTTP, a connection must be opened and closed for each Java class downloaded. The next generation HTTP (appropriately called HTTP-NG), should remove this considerable bottleneck[12]. The one-shot loading of Java is advantageous because it reduces the load on the server CPU and shifts the processing to the client. The server is therefore able to operate with a greater number clients. Once the applet is loaded on the client, it can operate independent of the server and network delays. The response times are dependent only on the local client..

Network loads are lowest utilizing Java because of the single communication between the client and server. Telnet's continuous connection and HTML's iterative communication have a larger load on the networks where they are being used. Both HTML and Java have a high level of user friendliness because of their ease of use. Each can utilize a graphical user interface (GUI) to create an intuitive interface. Both incorporate the mouse into their applications. The Telnet solution is strictly a text based interface and therefore tends to be more difficult to use. All three delivery methods are platform independent.

In conclusion, we propose the further research and eventual utilization of Java as a development environment for WWW CBPR. Its robust implementation will enable its ability to be used as an interactive tool in a clinical environment.

Acknowledgments

This work was supported in part by National Library of Medicine Training Grant LM07071-3. Additional thanks to Ahmed El-Ramly for the use of his computer.

References

1. Hinds A, Greenspun P, Kohane I S. WHAM!: Forms Constructor for Medical Record Access via the World Wide Web. In Gardner R, ed.: Proc of the 19th Annual SCAMC; New Orleans, LA; November, 1995:116-120.
2. Cimino J J, Socratous S A, Grewal R. The Informatics Superhighway: Prototyping on the World Wide Web. In Gardner R, ed.: Proc of the 19th Annual SCAMC; New Orleans, LA; November, 1995:111-115.
3. Socratous S A, Cimino J J, Clayton P D. Access to the Clincal Encounter via the World Wide Web (abstract). In Hripcsak G, ed.: Proc of the 1995 Spring Congress of AMIA; Boston, MA; June, 1995:85.
4. Cimino J J, Socratous S A, Clayton P D. Internet as clinical information system: application development utilizing the World Wide Web. JAMIA, 1995:2(5):273-284
5. Gosling J, McGilton H. The JavaTM Language: A White Paper, 1994, 1995. (see http://java.sun.com/whitePaper/).
6. Presently the operating systems include UNIX, Windows NT, Windows 95 and Macintosh PowerPC.
7. Campione, M, Walrath K. The Java Language Tutorial. (See http://java.sun.com/tutorial/ noMoreC/ index.html).
8. Netscape Corporation's Web Site. (See http://home.netscape.com/comprod/products/ Navigator/version_2.0/java_applets/index.html).
9. Earthweb, LLC. Gamelan Web Site. (See http://www.gamelan.com).
10. Daniel R. TeleMed GUI Applet. Los Alamos National Laboratory Web Site. (See http:// www.acl.lanl.gov/~rdaniel/classesJDK/ PickTesT2.html).
11. Stead WW, Hammond WE. Computer-based medical records: the centerpiece of TMR. MD Computing 1988;5(5):48-62.
12. World Wide Web Consortium Web Site. (See http:// www.w3.org/hypertext/WWW/Protocols/ HTTP-NG/ http-ng-arch.html).